Abbildung von Weltkoordinaten nach Bildkoordinaten

Werner Mayer

28. Februar 2004

Zusammenfassung

Dieses Dokument beschreibt die Abbildungsvorschrift von 3D-Punkten nach Pixelkoordinaten eines Bildes. Dabei wird zunächst der Aufbau der Matrizen für die Abbildung beschrieben und danach das Ausführen der Abbildung selbst. Zur Illustration ist am Ende noch ein kleines Beispiel in Maple angehängt.

1 Abbildung von Welt- in Bildkoordinaten

Die Abbildung von Welt- in Bildkoordinaten beinhaltet im Wesentlichen die Transformation von Weltkoordinaten in Kamerakoordinaten, die Projektion der Kamerakoordinaten in die Bildebene und letztlich die Skalierung auf die Bildgröße.

Die ersten beiden Umformungen lassen sich kompakt in einer 4x4-Matrix $\mathcal{M}$ beschreiben (siehe Abschnitt 2) und kann in einer Bilddatei abgelegt werden.

Die Skalierung auf die Bildgröße und damit die Berechnung der Bildkoordinaten wird in Abschnitt 3 näher beleuchtet.

2 Aufbau der 4x4-Matrix

Dieser Abschnitt beschreibt den Aufbau der 4x4-Matrix $\mathcal{M}$, die für die Transformation und Projektion der Punkte verantwortlich ist. Die Transformation und die Projektion lassen sich beide ebenfalls über die 4x4-Matrizen $\mathcal{T}$ und $\mathcal{P}$ formulieren. Mathematisch lässt sich die Gleichung

$$\mathcal{M} = \mathcal{P} \cdot \mathcal{T} \tag{1}$$

aufstellen. $\mathcal{M}$ kann dann in einer Datei abgespeichert werden, da diese alle relevanten Daten des Kameramodells enthält.

2.1 Transformationsmatrix

Dem Kameramodell müssen für den Aufbau der Transformationsmatrix $\mathcal{T}$ die folgenden Größen zu entnehmen sein:

- Up-Vektor Up
- Sichtvektor Dir
- Kameraposition P

Die hier beschriebene Transformation rechnet lediglich die Weltkoordinaten in Kamerakoordinaten um. Die Punkte werden an dieser Stelle noch nicht projiziert, daher spielt der Typ der Projektion (parallel, perspektiv) noch keine Rolle.
Mit den aus dem Kameramodell gewonnenen Größen lassen sich die Vektoren $\vec{u}$, $\vec{v}$, $\vec{w}$ und $\vec{p} \in \mathbb{R}^3$ wie folgt berechnen:

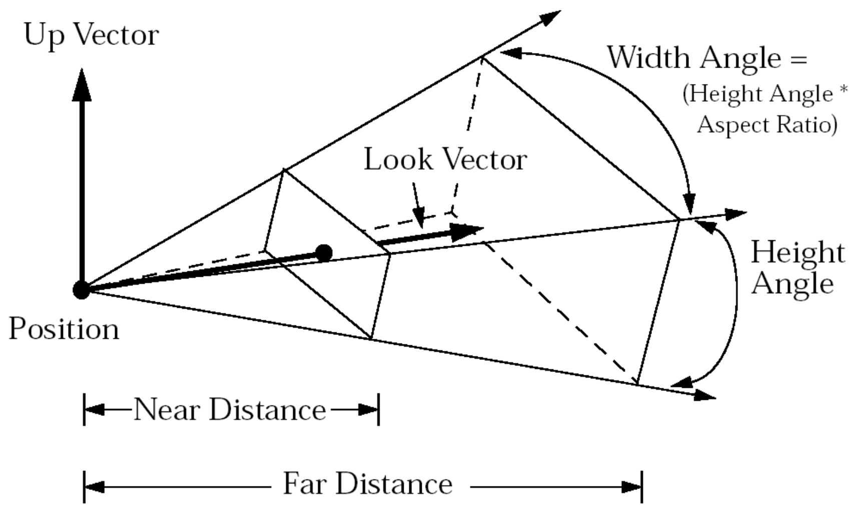


Abbildung 1: Die Abbildung zeigt ein Kameramodell – hier bei der perspektivischen Projektion.

$$
\begin{aligned}
\vec{u} &= Up \times (-Dir) = Dir \times Up \\
\vec{w} &= -Dir \\
\vec{v} &= w \times u \\
\vec{p} &= P
\end{aligned}
\tag{2}
$$

Aus (2) lässt sich dann die Transformationsmatrix $\mathcal{T}$ mit

$$
\mathcal{T} = \begin{pmatrix} u_x & u_y & u_z & -\vec{u}\cdot\vec{p} \\ v_x & v_y & v_z & -\vec{v}\cdot\vec{p} \\ w_x & w_y & w_z & -\vec{w}\cdot\vec{p} \\ 0 & 0 & 0 & 1 \end{pmatrix}
\tag{3}
$$

berechnen.

2.2 Projektionsmatrix

Dem Kameramodell müssen für den Aufbau der Projektionsmatrix $\mathcal{P}$ noch weiter die skalaren Größen l, r, b, t, n und f zu entnehmen sein. Je nach verwendetem Kameramodell kann es sein, dass eine Berechnung dieser Größen erforderlich ist, da diese möglicherweise nicht direkt zu entnehmen sind.

Bei der Berechnung der Projektionsmatrix $\mathcal{P}$ muss unterschieden werden, um welche Art von Projektion es sich handelt. Hier werden nur die *perspektivische* Projektion und die *Parallelprojektion* behandelt. Für die Berechnung werden ausschließlich die oben bestimmten Werte l, r, b, t, n und f verwendet.

2.2.1 Perspektivische Projektion

Bei der perspektivischen Projektion gilt für die Projektionsmatrix $\mathcal{P}$

$$\mathcal{P} = \begin{pmatrix} \frac{2\cdot n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2\cdot n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2\cdot f\cdot n}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{4}$$

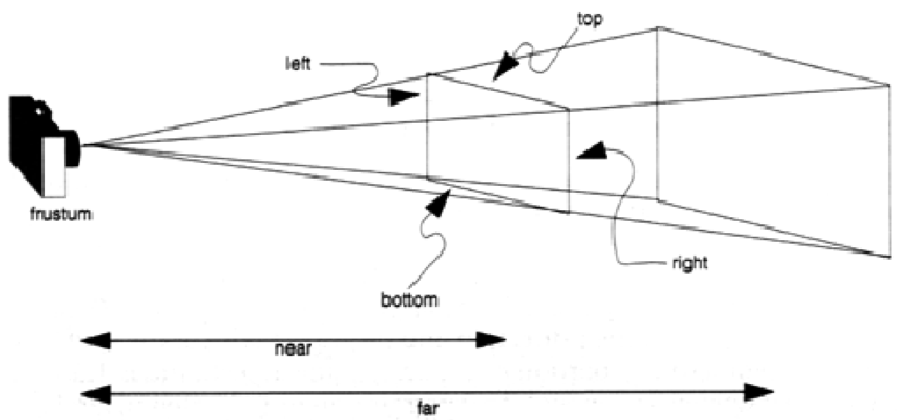


Abbildung 2: Die Abbildung zeigt das Viewing-Volume bei der perspektivischen Projektion - auch Frustum genannt.

2.2.2 Parallelprojektion

Bei der Parallelprojektion gilt für die Projektionsmatrix $\mathcal{P}$

$$\mathcal{P} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

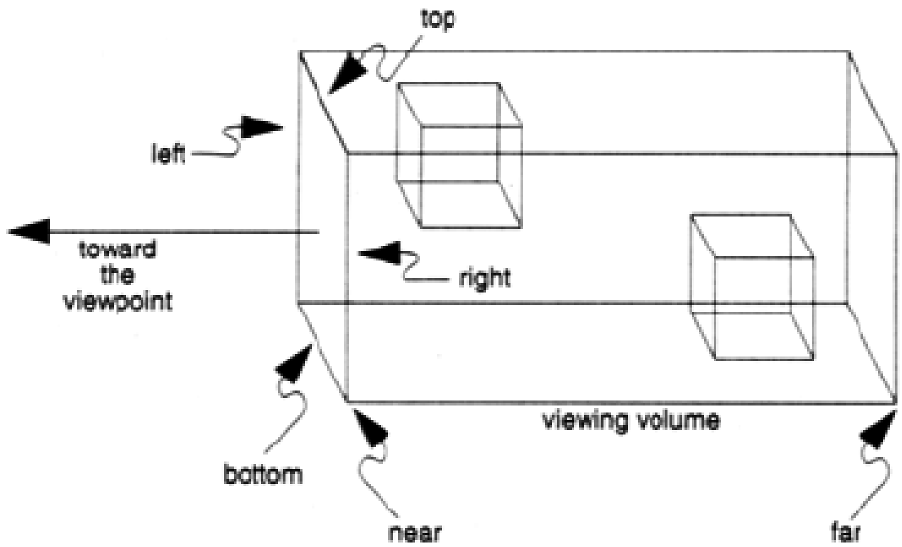


Abbildung 3: Die Abbildung zeigt das Viewing-Volume – einen Quader – bei der Parallelprojektion.

3 Skalierung

Dieser Abschnitt beschreibt, wie man letztlich von Weltkoordinaten zu Bildkoordinaten gelangt.

Dabei werden nur die in Abschnitt 2 bestimmte Matrix $\mathcal{M}$ sowie die Höhe h und die Breite b des Bildes benötigt

Ein Punkt $\mathbf{P} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) \in \mathbb{R}^3$ in Weltkoordinaten wird auf einen Punkt $\mathbf{Q} \in \mathbb{R}^3$ abgebildet, indem $\mathbf{P}$ mit $\mathcal{M}$ multipliziert wird.

$$
\begin{pmatrix} \mathbf{q}_x \\ \mathbf{q}_y \\ \mathbf{q}_z \\ w \end{pmatrix} = \mathcal{M} \cdot \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{p}_z \\ 1 \end{pmatrix} \tag{6}
$$

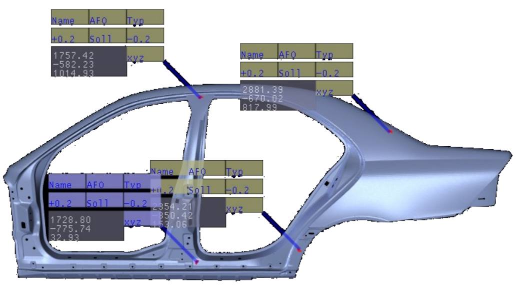
Schließlich erhält man mit

$$
k = \min\{b,\, h\}
$$

$$
\begin{pmatrix} \mathbf{s}_x \\ \mathbf{s}_y \end{pmatrix} = \frac{1}{2} \cdot \begin{pmatrix} k & 0 & b \\ 0 & -k & h \end{pmatrix} \cdot \begin{pmatrix} \mathbf{q}_x/w \\ \mathbf{q}_y/w \\ 1 \end{pmatrix} \tag{7}
$$

den Punkt $\mathbf{S} \in \mathbb{R}^2$ in Bildkoordinaten im Bild.

4 Beispiel




```
>  with(linalg):

>  M:=array(1..4,1..4,[ [2.40776,0.1705211,-0.04518837,-4849.866],
[-0.001357457,0.6363155,2.328847,-742.2269], [-0.6761357, 8.902349,
-2.432801, 9099.911], [-0.07306801,0.9620507,-0.2629057,5088.698]]);
```

$$M := \begin{bmatrix} 2.40776 & .1705211 & -.04518837 & -4849.866 \\ -.001357457 & .6363155 & 2.328847 & -742.2269 \\ -.6761357 & 8.902349 & -2.432801 & 9099.911 \\ -.07306801 & .9620507 & -.2629057 & 5088.698 \end{bmatrix}$$

```
> p:=array(1..4,[1757.42, -582.23, 1014.93, 1]);
```

$$p := [1757.42,\ -582.23,\ 1014.93,\ 1]$$

```
> multiply(M,p);
```

$$[-763.565953,\ 1248.522190,\ 259.319221,\ 4133.321157]$$

```
> q1 := -763.565953/4133.321157;
```

$$q1 := -.1847342425$$

```
> q2 := 1248.522190/4133.321157;
```

$$q2 := .3020627100$$

```
> S:=array(1..3,1..3,[[384,0,512],[0,-384,384], [0, 0, 1]]);
```

$$S := \begin{bmatrix} 384 & 0 & 512 \\ 0 & -384 & 384 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> q:=array(1..3,[q1, q2, 1]);
```

$$q := [-.1847342425,\ .3020627100,\ 1]$$

```
> multiply(S,q);
```

$$[441.0620509,\ 268.0079194,\ 1]$$